

# Compressive Sensing Using Low Density Frames

Mehmet Akçakaya, Jinsoo Park and Vahid Tarokh

**Abstract**—We consider the compressive sensing of a sparse or compressible signal  $\mathbf{x} \in \mathbb{R}^M$ . We explicitly construct a class of measurement matrices, referred to as the low density frames, and develop decoding algorithms that produce an accurate estimate  $\hat{\mathbf{x}}$  even in the presence of additive noise. Low density frames are sparse matrices and have small storage requirements. Our decoding algorithms for these frames have  $O(M)$  complexity. Simulation results are provided, demonstrating that our approach significantly outperforms state-of-the-art recovery algorithms for numerous cases of interest. In particular, for Gaussian sparse signals and Gaussian noise, we are within 2 dB range of the theoretical lower bound in most cases.

**Index Terms**—Low density frames, compressive sensing, sum product algorithm, expectation maximization, Gaussian scale mixtures

## I. INTRODUCTION

LET  $\mathbf{x} = (x_1, \dots, x_M) \in \mathbb{R}^M$  with  $\|\mathbf{x}\|_0 = |\{x_i : x_i \neq 0\}| = L$ .  $\mathbf{x}$  is said to be sparse if  $L \ll M$ . Consider the equation

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{n}, \quad (1)$$

where  $\mathbf{A}$  is an  $N \times M$  measurement matrix and  $\mathbf{n} \in \mathbb{R}^N$  is a noise vector. When  $N < M$ ,  $\mathbf{y}$  is called the compressively sensed version of  $\mathbf{x}$  with measurement matrix  $\mathbf{A}$ . In this paper, we are interested in coming up with a good estimate  $\hat{\mathbf{x}}$  for a sparse vector  $\mathbf{x}$  from the observed vector  $\mathbf{y}$  and the measurement matrix  $\mathbf{A}$ .

We refer to the case  $\mathbf{n} = \mathbf{0}$  as *noiseless compressive sensing*. This is the only case when  $\mathbf{x}$  can be perfectly recovered. In particular, it can be shown [13], [30] that if  $\mathbf{A}$  has the property that every of its  $N$  columns are linearly independent, then a decoder can recover  $\mathbf{x}$  uniquely from  $N = 2L$  samples by solving the  $\ell_0$  minimization problem

$$\min \|\mathbf{x}\|_0 \quad \text{s. t.} \quad \mathbf{y} = \mathbf{A}\mathbf{x}. \quad (2)$$

However, solving this  $\ell_0$  minimization problem for general  $\mathbf{A}$  is NP-hard [42]. An alternative solution method proposed in the literature is the  $\ell_1$  regularization approach, where

$$\min \|\mathbf{x}\|_1 \quad \text{s. t.} \quad \mathbf{y} = \mathbf{A}\mathbf{x}, \quad (3)$$

is solved instead. Criteria has been established in the literature as to when the solution of (3) coincides with that of (2) in the noiseless case for various classes of measurement matrices [13], [16]. In an important contribution, for  $\mathbf{A}$  belonging to the classes of Gaussian and partial Fourier ensembles, Candès and Tao showed [13] that this recovery problem can be solved for  $L = O(M)$  with  $N = O(L)$  as long as the observations are not contaminated with (additive) noise.

It can be shown that there is a relationship between the solution to problem (1) and minimum Hamming distance problem in coding theory [1], [2], [35]. This approach was further exploited in [50]. Using this connection, we constructed ensembles of measurement matrices<sup>1</sup> and associated decoding algorithms that solves the  $\ell_0$  minimization problem with complexity  $O(MN)$  for  $L = O(M)$  with  $N = O(L)$  in the noiseless case [1], [2].

For problem (1) with non-zero  $\mathbf{n}$ , referred to as *noisy compressive sensing*, the  $\ell_1$  regularization approach of (3) can also be applied. For a measurement matrix  $\mathbf{A}$  that satisfies a property called the restricted isometry principle (RIP), the quadratic program

$$\min \|\mathbf{x}\|_1 \quad \text{s. t.} \quad \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2 \leq \epsilon,$$

can be solved for a parameter  $\epsilon$  related to  $\|\mathbf{n}\|_2$ , and an estimate  $\hat{\mathbf{x}}_{\text{QP}}$  can be obtained such that  $\|\hat{\mathbf{x}}_{\text{QP}} - \mathbf{x}\|_2 \leq C_1 \epsilon$ , where  $C_1$  is a constant that depends on  $\mathbf{A}$  [11]. If  $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_N)$ , another approach is the Dantzig Selector

$$\min \|\mathbf{x}\|_1 \quad \text{s. t.} \quad \|\mathbf{A}^*(\mathbf{A}\mathbf{x} - \mathbf{y})\|_\infty \leq \gamma,$$

where  $\gamma$  is a function of  $\sigma$  and  $M$ . This gives an estimate  $\hat{\mathbf{x}}_{\text{DS}}$  such that  $\mathbb{E}_{\mathbf{n}} \|\hat{\mathbf{x}}_{\text{DS}} - \mathbf{x}\|_2^2 \leq C_2 (\log M) \sum_i \min(x_i^2, \sigma^2)$ , where  $C_2$  is a constant that depends on  $\mathbf{A}$  [12]. Both these methods may not be easily implemented in real time with the limitations of today's hardware. To improve the running time of  $\ell_1$  methods, some authors have investigated using sparse matrices for  $\mathbf{A}$  [6]. Using the expansion properties of the graphs represented by such matrices, it was shown in [6] that it is possible to obtain an estimate  $\hat{\mathbf{x}}_{\text{E}}$  such that  $\|\hat{\mathbf{x}}_{\text{E}} - \mathbf{x}\|_1 \leq C_3 \|\mathbf{n}\|_1$ , where  $C_3$  is a constant that depends on  $\mathbf{A}$ .

Another strand of work studies recovery algorithms based on the matching pursuit algorithm [44]. Recently, variants of this algorithm, e.g. Subspace Pursuit [17] and CoSaMP [30], have been proposed. Both algorithms provably work for measurement matrices satisfying RIP, and guarantee perfect reconstruction in the noiseless setting for  $N = O(L \log(M/L))$  as the  $\ell_1$  recovery methods do. For the noisy problem, they also offer similar guarantees to  $\ell_1$  methods. These algorithms have complexity  $O(\mathcal{L} \log R)$ , where  $\mathcal{L}$  is the complexity of matrix-vector multiplication ( $O(MN)$  for Gaussian matrices,  $O(N \log N)$  for partial Fourier ensembles) and  $R$  is a precision parameter bounded above by  $\|\mathbf{x}\|_2$  (which is  $O(N)$  for a fixed signal-to-noise ratio). In [7], Sparse Matching Pursuit (SMP) was proposed for sparse  $\mathbf{A}$  and this algorithm has  $O(M \log(M/L))$  complexity.

M. Akçakaya, J. Park and V. Tarokh are with the School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, 02138. (e-mails: {akcakaya, vahid}@seas.harvard.edu, park10@fas.harvard.edu)

<sup>1</sup>We use the terms “frame” and “measurement matrix” interchangeably throughout the rest of the paper.

Yet another direction in compressive sensing is the use of the Bayesian approach. In [23], the idea of relevance vector machine (RVM) [40] has been applied to compressive sensing. Although simulation results indicate that the algorithm has good performance, it has complexity  $O(MN^2)$ .

In this paper, we study the construction of measurement matrices that can be stored and manipulated efficiently in high dimensions, and fast decoding algorithms that generate estimates with small  $\ell_2$  distortion. The ensemble of measurement matrices are a generalization of LDPC codes and we refer to them as *low density frames* (LDFs). For our decoding algorithms, we combine various ideas from coding theory, statistical learning theory and theory of estimation. Simulation results are provided indicating an excellent distortion performance at high levels of sparsity and for high levels of noise.

The outline of this paper is given next. In Section II, we introduce low density frames and study their basic properties. In Section III, we introduce various concepts used in algorithms and describe the decoding algorithms. In Section IV, we provide extensive simulation results for a number of different testing criteria. Finally in Section V, we make our conclusions and provide directions for future research.

## II. LOW DENSITY FRAMES

Let  $\mathcal{F} = \{\phi_1, \phi_2, \dots, \phi_M\}$  be a frame consisting of  $M \geq N$  non-zero vectors which span  $\mathbb{R}^N$ . Let  $\phi_i = (\phi_{1,i}, \dots, \phi_{N,i})$  for  $i = 1, 2, \dots, M$ . This frame could be represented in matrix form as an  $N \times M$  matrix

$$\mathbf{F} = \begin{pmatrix} \phi_{1,1} & \phi_{1,2} & \cdots & \phi_{1,M} \\ \phi_{2,1} & \phi_{2,2} & \cdots & \phi_{2,M} \\ \vdots & \ddots & \ddots & \vdots \\ \phi_{N,1} & \phi_{N,2} & \cdots & \phi_{N,M} \end{pmatrix}. \quad (4)$$

A *low density frame* (LDF)  $\mathcal{F}$  is defined by a matrix  $\mathbf{F}$  where the vast majority of elements of each column and each row of  $\mathbf{F}$  are zeroes. Formally, we define a  $(d_v, d_c)$ -regular LDF as a matrix  $\mathbf{F}$  that has  $d_c$  non-zero elements in each row and  $d_v$  non-zero elements in each column. Clearly  $Md_v = Nd_c$ . We also note that the redundancy of the frame is  $r = M/N = d_c/d_v$ . We will restrict ourselves to *binary* regular LDFs, where the non-zero elements of  $\mathbf{F}$  are ones.

The density  $\rho$  of a frame  $\mathbf{F}$  is the ratio of the number of non-zero entries of  $\mathbf{F}$  to the dimension of  $\mathbf{F}$ . In this paper, we consider regular LDFs for which  $\rho = (Md_v)/(MN) = d_v/N < 1$ .

As with LDPC codes, it is natural to represent LDFs using bipartite graphs. Furthermore, there is a well-established literature on inference in graphical models. Some of these methods can be used as a basis for recovery algorithms in the context of compressive sensing. To this end, we next summarize two important ideas from graphical models, namely factor graphs and the sum-product algorithm, and show how LDFs can be viewed as factor graphs.

### A. Factor Graphs

Factor graphs are used to represent factorizations of functions of several variables [9], [24]. Let  $f(\mathbf{w})$  be a function of

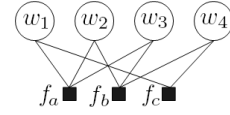


Fig. 1. Factor graph representing the function in Equation 6.

several variables that can be factored as

$$f(\mathbf{w}) = \prod_s f_s(\mathbf{w}_s). \quad (5)$$

In this factorization each *factor*  $f_s$  is only a factor of  $\mathbf{w}_s$ , the subset of *variable nodes*  $\mathbf{w}$ .

A *factor graph* depicting (5) consists of variable nodes represented by circles, factor nodes represented by bold squares, and undirected edges connecting each factor node to all the variable nodes involved in that factor.

As an example, the factor graph representing

$$f(\mathbf{w}) = f_a(w_1, w_2, w_3) f_b(w_2, w_3, w_4) f_c(w_1, w_4) \quad (6)$$

is depicted in Figure 1.

### B. Sum-Product Algorithm

The natural inference algorithm for factor graphs is the sum-product algorithm [9], [21]. This algorithm is an exact interference algorithm for tree-structured graphs (i.e. graphs with no cycles), and is usually described over discrete alphabets. However, the ideas also apply to continuous random variables with the sum being replaced by integration. In doing so, the computational cost of implementation increases and this issue will be addressed later.

Suppose the goal is to find the marginal density  $p(w)$  for a particular variable  $w$ . In particular, we have

$$p(w) = \sum_{\mathbf{w} \setminus w} p(\mathbf{w}).$$

One treats  $w$  to be the root node of a tree, and looks at the subtrees connected to  $w$  via factor nodes. Using this approach, the joint distribution can be written as

$$p(\mathbf{w}) = \prod_{s \in ne(w)} F_s(w, W_s), \quad (7)$$

where  $ne(w)$  is the neighborhood of  $w$ , i.e. the set of factor nodes that are connected to  $w$ , and  $W_s$  is the set of variable nodes in the subtree connected to the factor node  $f_s$  in  $ne(w)$  [9].  $F_s(w, W_s)$  represents the product of the factors in the subtree associated with  $f_s$ . Interchanging the summation and the products yields

$$p(w) = \prod_{s \in ne(w)} \mu_{f_s \rightarrow w}(w),$$

where  $\mu_{f_s \rightarrow w}(w)$  is the message sent from factor node  $f_s$  to variable node  $w$ . One can show [9] that

$$\mu_{f_s \rightarrow w}(w) = \sum_{\mathbf{w}_s \setminus w} f_s(w, \mathbf{w}_s) \prod_{m \in ne(f_s) \setminus w} \mu_{w_m \rightarrow f_s}(w_m),$$

where  $\mathbf{w}_s$  are all variable nodes connected to the factor node  $f_s$ , including  $w$ , and  $ne(f_s)$  are the set of variable nodes connected to the factor node  $f_s$ . One can also show [9]

$$\mu_{w_m \rightarrow f_s}(w_m) = \prod_{l \in ne(w_m) \setminus f_s} \mu_{f_l \rightarrow w_m}(w_m).$$

Thus there are two types of messages, one type going from factor nodes to variable nodes denoted  $\mu_{f \rightarrow w}$  and the other going from variable nodes to factor nodes denoted  $\mu_{w \rightarrow f}$ . The message propagation starts from the leaves of the factor graph. A leaf variable node sends an identity message  $\mu_{w \rightarrow f}(w) = 1$  to its parent, whereas a leaf factor node  $f$  sends  $\mu_{f \rightarrow w}(w) = f(w)$ , a description of  $f$  to its parent. These expressions for messages give an efficient way of calculating the marginal probability distribution. We note that in writing out the factorization in (7), it is essential that the graph has a tree structure so that the factors in the joint probability distribution  $p(\mathbf{w})$  can be partitioned into groups, each of which is associated with a single factor node in  $ne(w)$ .

The algorithm is easily modified to calculate the marginal for every variable node in the graph [9]. This modification results in only twice as many calculations as calculating a single marginal. A more interesting case is when there are observed variables in the graph,  $\mathbf{v}$ . In this case the sum-product algorithm could be used to calculate posterior marginals  $p(w_i | \mathbf{v} = \hat{\mathbf{v}})$ .

### C. Graphical Representation of Low Density Frames

The main connection between the  $\ell_0$  minimization problem and coding theory involves the description of the underlying code [1],  $\mathcal{V}$  of  $\mathbf{F}$ , where

$$\mathcal{V} = \{\mathbf{d} \in \mathbb{R}^M : \mathbf{F}\mathbf{d} = \mathbf{0}\}.$$

One can view  $\mathcal{V}$  as the set of vectors whose product with each row of  $\mathbf{F}$  “checks” to 0. In the works of Tanner, it was noted that this relationship between the checks and the codewords of a code can be represented by a bipartite graph [39]. This bipartite graph consists of two disjoint sets of vertices,  $V$  and  $C$ , where  $V$  contains the variable nodes and  $C$  contains the factor nodes representing checks that codewords need to satisfy. Thus we have  $|V| = M$  and  $|C| = N$ . Furthermore node  $j$  in  $V$  will be connected to node  $i$  in  $C$  if and only if the  $(i, j)^{\text{th}}$  element of  $\mathbf{F}$  is non-zero. Thus the number of edges of the graph is equal to the number of non-zero elements in the measurement matrix  $\mathbf{F}$ . For an LDF, this leads to a sparse bipartite graph.

A simple example of this graphical representation is depicted in Figure 2. For representation of LDFs, it is convenient to use a factor node, depicted by a  $\boxplus$ , called a parity check node. This node has the property that the variable nodes connected to it should sum to zero. We also note that for the purposes of decoding, it is more convenient to use syndromes [26] that represent the measurement vector,  $\mathbf{r}$ . This is done by connecting a variable node representing the  $j^{\text{th}}$  component of  $\mathbf{r}$  to the  $j^{\text{th}}$  check node. In this case, the parity check node has the property that the variable nodes connected to it sum to  $r_j$ . Thus the graph now represents the set  $\{\mathbf{d} \in \mathbb{R}^M : \mathbf{F}\mathbf{d} = \mathbf{r}\}$  which is a coset of the underlying code of the frame.

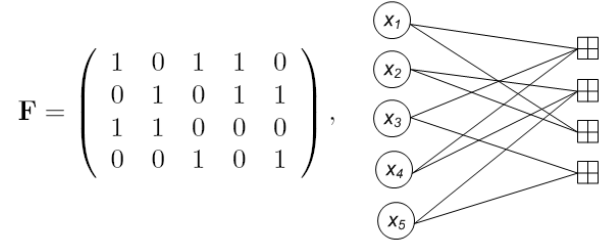


Fig. 2. A frame  $\mathbf{F}$  and its graphical representation.

It is important to note that the graph representing an LDF will have cycles. Without the tree structure, the sum-product algorithm will only be an approximate inference algorithm. However, it has been empirically shown that for sparse graphs this approximate algorithm works very well [25], [33], [48].

### III. SUM PRODUCT ALGORITHM WITH EXPECTATION MAXIMIZATION

It is well-known in coding theory literature, that the standard decoding algorithm for codes on graphs is the sum-product algorithm (SPA) [21], [24], [25]. Given a set of observations, this algorithm can be used to approximate the posterior marginal distributions. In fact, when there is no noise, variants of this algorithm [38] has been successfully adapted to compressive sensing [35], [50]. However, when we are interested in the practical case of noisy observations, these algorithms no longer can be applied in a straightforward manner. Some authors have tried to circumvent this difficulty by using a two-point Gaussian mixture approach [36], however the complexity of this algorithm may grow quickly as the number of Gaussian components in the mixtures could grow exponentially, unless some approximation is applied. However, using these approximations degrades the performance of the LDF approach.

In this paper, we consider Gaussian Scale Mixture (GSM) priors with Jeffreys’ non-informative hyperprior to obtain an algorithm that provides estimates for the noisy compressive sensing problem

$$\mathbf{r} = \mathbf{F}\mathbf{x} + \mathbf{n},$$

as well as the noiseless problem. Throughout the paper we assume that

$$\mathbf{n} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_N).$$

However simulation results (not included in this paper) indicate that the algorithms still work well even for non-Gaussian noise. We define the signal-to-noise ratio (SNR) as

$$\text{SNR} = 10 \log_{10} \frac{\|\mathbf{F}\mathbf{x}\|^2}{\mathbb{E}_{\mathbf{n}} \|\mathbf{n}\|^2} = 10 \log_{10} \frac{\|\mathbf{F}\mathbf{x}\|^2}{\sigma^2 N}.$$

#### A. Gaussian Scale Mixtures

The main difficulty in using the sum-product algorithm (SPA) in compressive sensing setting is that the variables of interest are continuous. Nonetheless SPA can be employed naturally when the underlying continuous random variables are Gaussian [47]. Since any Gaussian pdf  $\mathcal{N}(x|a, A)$  can be

determined by its mean  $a$  and variance  $A$ , these constitute the messages in this setting. At the variable nodes, the product of Gaussian probability density functions (pdf) will result in a (scaled) Gaussian pdf, and at the check nodes, the convolution of Gaussian pdfs will also result in a Gaussian pdf. i.e.

$$\mathcal{N}(x|a_1, A_1) * \mathcal{N}(x|a_2, A_2) \propto \mathcal{N}(x|a_1 + a_2, A_1 + A_2),$$

and

$$\mathcal{N}(x|a_1, A_1) \cdot \mathcal{N}(x|a_2, A_2) \propto \mathcal{N}(x|b, B),$$

where  $\propto$  denotes normalization up to a constant, and

$$B = (A_1^{-1} + A_2^{-1})^{-1},$$

$$b = B(A_1^{-1}a_1 + A_2^{-1}a_2).$$

We note that all the underlying operations for SPA preserve the Gaussian structure.

It is well-known that the Gaussian pdf is not “sparsity-enhancing”. Thus some authors propose the use of the Laplacian prior

$$p(\mathbf{x}) = \prod_i p_{x_i}(x_i) = \prod_i \frac{\lambda}{2} \exp(-\lambda|x_i|). \quad (8)$$

Clearly with this prior and for Gaussian noise  $\mathbf{n}$

$$p(\mathbf{x}|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) \propto \exp(-\|\mathbf{y} - \mathbf{Ax}\|_2^2 - \lambda'\|\mathbf{x}\|_1), \quad (9)$$

and maximization of  $p(\mathbf{x}|\mathbf{y})$  is equivalent to minimizing

$$\|\mathbf{y} - \mathbf{Ax}\|_2^2 + \lambda'\|\mathbf{x}\|_1,$$

which is the objective function for the LASSO algorithm [43], [46]. However, a straightforward implementation of this algorithm may not be computationally feasible.

In this paper, we consider the family of Gaussian Scale Mixtures (GSM) densities [4], given by

$$x = \sqrt{\beta}u,$$

where  $u$  is a zero-mean Gaussian and  $\sqrt{\beta}$  is a positive scalar random variable. Hence

$$p_{x|\beta}(x|\beta) \sim \mathcal{N}(x|0, \beta),$$

and

$$p_x(x) = \int_0^\infty p_{x|\beta}(x|\beta)p_\beta(\beta)d\beta.$$

This family of densities are symmetric, zero-mean and have heavier tails than a Gaussian, and have been successfully used in image processing [8], [18], [32], and learning theory [40].

In order to completely specify our model, we need to choose a pdf for  $p_\beta(\beta)$ . In this paper, we use

$$p_\beta(\beta) \propto \sqrt{\det(I(\beta))}, \quad I(\beta) = \mathbb{E}\left(-\frac{\partial^2 \log p_{x|\beta}(x|\beta)}{\partial \beta^2} \middle| \beta\right)$$

where  $I(\beta)$  is the Fisher information matrix. This is referred to as the Jeffreys' prior, which can be shown to be a scalar invariant prior suitable for sparse estimation [34]. In our model, the prior is given by

$$p_{\beta_i}(\beta_i) = \frac{1}{\beta_i},$$

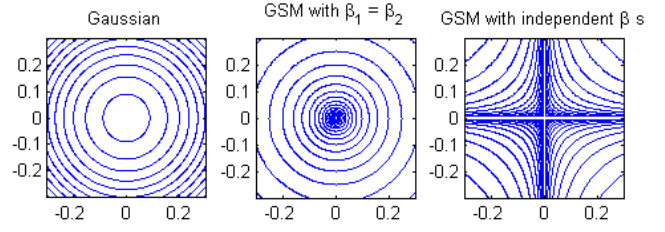


Fig. 3. Contour plots for a Gaussian distribution (left), a GSM with  $\beta_1 = \beta_2$  distributed according to Jeffreys' prior (middle), a GSM with  $\beta_1$  and  $\beta_2$  i.i.d. with Jeffreys' prior (right).

which has no parameters to optimize. We note that this is an improper density, i.e. it cannot be normalized. In Bayesian statistics, these kind of improper priors are used frequently, since only the relative weight of the prior determines the a-posteriori density [34]. This density also has a singularity at the origin. This fact is usually ignored as long as it does not create computational problems [32]. As an alternative one might set the prior to 0 in a small interval  $\beta \in [0, \beta_{min})$ . We also note that with this choice for  $p_{\beta_i}(\beta_i)$ ,  $p_{x_i}(x_i) \propto 1/|x_i|$ , which is a very heavy-tailed density.

To enhance sparsity in each coordinate, it is important to have independent  $\beta_i$  for all  $i$  [41]. As depicted in the middle subplot of Figure 3, compared to a Gaussian distribution, a GSM with  $\beta_i$  distributed according to Jeffreys' prior has a much sharper peak at the origin even when  $\beta_1 = \beta_2$ . However, the subplot on the right demonstrates that if the  $\beta_i$ s are indeed independent, the GSM will be highly concentrated not only around the origin, but along the coordinate axes as well, which is a desired property if we have no further information about the locations of the sparse coefficients of  $\mathbf{x}$ . In our model, we will assume that

$$p(\mathbf{x}, \boldsymbol{\beta}) = \prod_{i=1}^M p(x_i|\beta_i) \prod_{i=1}^M p(\beta_i)$$

in order to enhance sparsity in all coordinates. This independence assumption is natural and commonly used in the literature [18], [43], [46].

### B. Expectation Maximization

The expectation maximization (EM) algorithm is a method for finding maximum-likelihood (ML) estimates of parameters in a model with observed and hidden variables [29]. Let  $\mathbf{y}$  be the observed data and let  $\mathbf{z}$  be the hidden data. Let the probability density function of  $(\mathbf{y}, \mathbf{z})$  be  $f(\mathbf{y}, \mathbf{z}|\boldsymbol{\theta})$ , parametrized by the vector  $\boldsymbol{\theta}$ . The EM algorithm iteratively improves on an initial estimate  $\boldsymbol{\theta}^{(0)}$  using a two-step procedure. In the expectation step (E-step), we calculate

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(k)}) = \mathbb{E}_{\mathbf{z}}\left(\log f(\mathbf{y}, \mathbf{z}|\boldsymbol{\theta}) \middle| \mathbf{y}, \boldsymbol{\theta}^{(k)}\right)$$

given an estimate  $\boldsymbol{\theta}^{(k)}$  from the previous iteration. It is important to distinguish the two arguments of the  $Q$  function are different. The second argument is the conditioning argument for the expectation and is fixed during the E-step. In the second

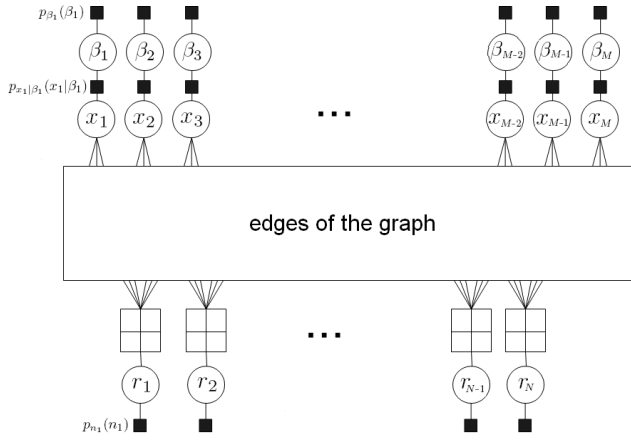


Fig. 4. The factor graph for a (3,6)-regular LDF with the appropriate hyperpriors.

step, called the maximization step or M-step, a new estimate

$$\theta^{(k+1)} = \arg \max_{\theta} Q(\theta | \theta^{(k)})$$

is calculated.

It can be shown that the estimates monotonically increases the likelihood with respect to the observed data  $\mathbf{y}$  [29],

$$f(\mathbf{y} | \theta^{(k+1)}) \geq f(\mathbf{y} | \theta^{(k)}).$$

When  $\theta$  is itself a random variable, the M-step maximizes  $(Q(\theta | \theta^{(k)}) + \log f(\theta))$ , and the EM algorithm can be used to find a maximum a-posteriori (MAP) estimate of  $\theta$  [28].

### C. SuPrEM Algorithm I

The factor graph for decoding purposes is depicted in Figure 4. Here,  $\mathbf{r}$  is the vector of observed variables,  $\mathbf{x}$  is the vector of hidden variables and  $\beta$  is the vector of parameters. We next propose the *Sum Product with Expectation Maximization* (SuPrEM) Algorithm I. At every iteration  $t$ , this algorithm uses a combination of the Sum-Product Algorithm (SPA) and EM algorithm to generate estimates for the hyperpriors  $\{\beta_k^{(t)}\}$ , as well as a point estimate  $\{\hat{x}_k^{(t)}\}$ . In the EM stage of the algorithm,  $Q(\beta | \beta^{(t)})$  for the **E-step** is given by

$$\begin{aligned} Q(\beta | \beta^{(t)}) &= \mathbb{E}_{\mathbf{x}} \left( \log p(\mathbf{x}, \mathbf{y}, \beta) | \mathbf{y}, \beta^{(t)} \right) \\ &= \mathbb{E}_{\mathbf{x}} \left( \log (p(\mathbf{y} | \mathbf{x}) p(\mathbf{x} | \beta) p(\beta)) | \mathbf{y}, \beta^{(t)} \right) \\ &= \mathbb{E}_{\mathbf{x}} \left( \log p(\mathbf{y} | \mathbf{x}) | \mathbf{y}, \beta^{(t)} \right) + \mathbb{E}_{\mathbf{x}} \left( \log p(\mathbf{x}, \beta) | \mathbf{y}, \beta^{(t)} \right) \\ &= C_1 + \sum_{i=1}^M \mathbb{E}_{\mathbf{x}} \left( \log p(x_i, \beta_i) | \mathbf{y}, \beta^{(t)} \right) \\ &= C_1 + \sum_{i=1}^M \mathbb{E}_{x_i} \left( \log p(x_i, \beta_i) | \mathbf{y}, \beta^{(t)} \right), \end{aligned} \quad (10)$$

where  $C_1 = \mathbb{E}_{\mathbf{x}} (\log p(\mathbf{y} | \mathbf{x}) | \mathbf{y}, \beta^{(t)})$  is a term independent of  $\beta$ . Let  $Q(\beta_i | \beta^{(t)}) = \mathbb{E}_{x_i} (\log p(x_i, \beta_i) | \mathbf{y}, \beta^{(t)})$ . We have

$$Q(\beta | \beta^{(t)}) = C_1 + \sum_{i=1}^M Q(\beta_i | \beta^{(t)}) \quad (11)$$

Since in our setting, the underlying variables are Gaussian, the density  $p(x_i | \mathbf{y}, \beta^{(t)})$  produced by the SPA is also Gaussian, with mean  $\mu_i$  and variance  $\nu_i$ . One can explicitly write out  $Q(\beta_i | \beta^{(t)})$  as

$$\begin{aligned} Q(\beta_i | \beta^{(t)}) &= \mathbb{E}_{x_i} \left( \log p(x_i, \beta_i) | \mathbf{y}, \beta^{(t)} \right) \\ &= \mathbb{E}_{x_i} \left( \log \left( \frac{1}{\sqrt{2\pi}\beta_i} \exp\left(-\frac{x_i^2}{2\beta_i}\right) \frac{1}{\beta_i} \right) | \mathbf{y}, \beta^{(t)} \right) \\ &= C_2 - \frac{3}{2} \log \beta_i - \frac{1}{2\beta_i} \mathbb{E}_{x_i} (x_i^2 | \mathbf{y}, \beta^{(t)}) \\ &= C_2 - \frac{3}{2} \log \beta_i - \frac{1}{2\beta_i} (\mu_i^2 + \nu_i), \end{aligned} \quad (12)$$

where  $C_2$  is independent of  $\beta_i$ .

For the **M-step**, we find

$$\beta^{(t+1)} = \arg \max_{\beta} Q(\beta | \beta^{(t)}).$$

Clearly  $Q(\beta | \beta^{(t)})$  can be maximized by maximizing each  $Q(\beta_i | \beta^{(t)})$ . Hence we have the simple local update rule

$$\beta_i^{(t+1)} = \arg \max_{\beta_i} Q(\beta_i | \beta^{(t)}) = \frac{\mu_i^2 + \nu_i}{3} \quad (13)$$

We summarize SuPrEM I in Algorithm 1. The inputs to the algorithm contain a stopping criterion  $\mathcal{T}$  and a message-passing schedule  $\mathcal{S}$ . The stopping criterion does not really affect the behavior of the algorithm, and there are a few alternatives for a reasonable criterion, which are discussed in Section IV. It turns out the message-passing schedule is rather important for achieving the maximum performance. To this end, we develop a message-passing schedule that attains such good performance and we describe this schedule in detail in Appendix I. For all our simulations, we use this fixed schedule. Simulation results indicate that with this fixed schedule, the algorithm is robust in various different scenarios. The overall complexity of SuPrEM is  $O(M)$  for a fixed number of iterations. We also note that in the presence of noise, the output of the algorithm will not be exactly sparse and a sparse estimate can be constructed using soft-thresholding techniques such as those described in [18].

### D. SuPrEM Algorithm II

When the ratio  $L/N$  is relatively large, SuPrEM I does not perform well, in particular for high SNRs, since it does not enforce strict sparsity. Thus we propose SuPrEM Algorithm II that enforces sparsity at various stages of the algorithm and sends messages between the nodes of the underlying graph accordingly. To this end, we keep a set of candidate variable nodes  $\mathcal{O}$  that are likely to have non-zero values, and modify the messages from the variable nodes that do not belong to a specified subset of  $\mathcal{O}$  denoted by  $\mathcal{O}_1$ . Similar ideas have been used in developing state-of-the-art recovery algorithms for compressive sensing, such as Subspace Pursuit [17] and CoSaMP [30]. The full description is given in Algorithm 2.

The main modification to SuPrEM I is the addition of a sparsification step. Intuitively  $\beta_k^{(t)}$  is the reliability of the hypothesis  $\hat{x}_k^{(t)} \neq 0$ . Throughout the algorithm we maintain

---

**Algorithm 1** SuPrEM Algorithm I
 

---

**Inputs:** The observed vector  $\mathbf{r}$ , the measurement matrix  $\mathbf{F}$ , the noise level  $\sigma^2$ , a stopping criterion  $\mathcal{T}$ , and a message-passing schedule  $\mathcal{S}$ .

**1. Initialization:** Let  $\beta_k^{(0)} = |(\mathbf{F}^T \mathbf{r})_k|^2 / d_v^2$ . Initial outgoing messages from variable node  $x_k$  is  $(0, \beta_k^{(0)})$ .

**2. Check Nodes:** For  $i = 1, 2, \dots, N$

Let  $\{i_1, i_2, \dots, i_{d_c}\}$  be the indices of the variable nodes connected to the  $i^{\text{th}}$  check node  $r_i$ . Let the message coming from variable node  $x_{i_j}$  to the check node  $r_i$  at  $t^{\text{th}}$  iteration be  $(\mu_{i_j}^{(t)}, \nu_{i_j}^{(t)})$  for  $j = 1, \dots, d_c$ . Then the outgoing message from check node  $r_i$  to variable node  $x_{i_j}$  is

$$(r_i - \sum_{k=1, k \neq j}^{d_c} \mu_{i_k}^{(t)}, \sum_{k=1, k \neq j}^{d_c} \nu_{i_k}^{(t)} + \sigma^2).$$

The messages are sent according to the schedule  $\mathcal{S}$ .

**3. Variable Nodes:** For  $k = 1, 2, \dots, M$

Let  $\{k_1, k_2, \dots, k_{d_v}\}$  be the indices of the check nodes connected to the  $k^{\text{th}}$  variable node  $x_k$ . Let the incoming message from the check node  $r_{k_j}$  to the variable node  $x_k$  at the  $t^{\text{th}}$  iteration be  $(\mu_{k_j}^{(t)}, \nu_{k_j}^{(t)})$  for  $j = 1, \dots, d_v$ .

**a. EM update:** Let

$$V_k^{(t)} = \left( \sum_{j=1}^{d_v} \frac{1}{\nu_{k_j}^{(t)}} + \frac{1}{\beta_k^{(t-1)}} \right)^{-1}, \mu_k^{(t)} = V_k^{(t)} \left( \sum_{j=1}^{d_v} \frac{\mu_{k_j}^{(t)}}{\nu_{k_j}^{(t)}} \right).$$

Then the EM update is  $\beta_k^{(t)} = \frac{(\mu_k^{(t)})^2 + V_k^{(t)}}{3}$ .

**b. Message updates:** The outgoing message from variable node  $x_k$  to check node  $r_{k_i}$  at the  $(t+1)^{\text{th}}$  iteration is given by  $(\mu_{k_i}^{(t+1)}, \nu_{k_i}^{(t+1)})$ , where

$$\nu_{k_i}^{(t+1)} = \left( \sum_{j=1, j \neq i}^{d_v} \frac{1}{\nu_{k_j}^{(t)}} + \frac{1}{\beta_k^{(t)}} \right)^{-1}$$

and

$$\mu_{k_i}^{(t+1)} = \nu_{k_i}^{(t+1)} \left( \sum_{j=1, j \neq i}^{d_v} \frac{\mu_{k_j}^{(t)}}{\nu_{k_j}^{(t)}} \right).$$

The messages are sent according to the schedule  $\mathcal{S}$ .

**4. Iterations:** Repeat (2) and (3) until stopping criterion  $\mathcal{T}$  is reached.

**5. Decisions:** For the  $k^{\text{th}}$  variable node  $x_k$ , let the incoming messages be  $(\mu_{k_j}^{(\mathcal{T})}, \nu_{k_j}^{(\mathcal{T})})$  for  $j = 1, \dots, d_v$ . Let

$$\hat{V}_k = \left( \sum_{j=1}^{d_v} \frac{1}{\nu_{k_j}^{(\mathcal{T})}} + \frac{1}{\beta_k^{(\mathcal{T})}} \right)^{-1}$$

and

$$\hat{x}_k = \hat{V}_k \left( \sum_{j=1}^{d_v} \frac{\mu_{k_j}^{(\mathcal{T})}}{\nu_{k_j}^{(\mathcal{T})}} \right).$$

**Output:** The estimate is  $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_M)^T$ .

---

a list of variable nodes  $\mathcal{O}_1$  that correspond to the largest  $L$  coefficients of  $\hat{\mathbf{x}}^{(t)}$  at iteration  $t$ . We also keep a list of variable nodes  $\mathcal{O}_2$  corresponding to the  $L$  largest elements of  $\beta^{(t)}$ , i.e. those with the largest reliabilities of the hypothesis  $\hat{x}_k^{(t)} \neq 0$ . In the sparsification stage, these two sets are merged,  $\mathcal{O} = \mathcal{O}_1 \cup \mathcal{O}_2$ . The addition and deletion of elements from  $\mathcal{O}$  allow refinements to be made with each iteration. We note  $L \leq |\mathcal{O}| \leq 2L$  at any given iteration. Decisions are made on the elements of  $\mathcal{O}$ , and  $\mathcal{O}_1$  is updated. Finally for variable nodes not in  $\mathcal{O}_1$ , the mean value of the messages is forced to

---

**Algorithm 2** SuPrEM Algorithm II
 

---

**Inputs:** The observed vector  $\mathbf{r}$ , the measurement matrix  $\mathbf{F}$ , the sparsity level  $L$ , a stopping criterion  $\mathcal{T}$ , the noise level  $\sigma^2$  (optional), and a message-passing schedule  $\mathcal{S}$ .

**1. Initialization:** Let  $\beta_k^{(0)} = |(\mathbf{F}^T \mathbf{r})_k|^2 / d_v^2$  and let  $\mathcal{O}_1 = \emptyset$ . Initial outgoing messages from variable node  $x_k$  is  $(0, \beta_k^{(0)})$ .

**2. Check Nodes:** Same as in Algorithm I.

**3. Variable Nodes:** Same as in Algorithm I.

**4. Sparsification:**

**a.** After the  $\beta_k$ s have been updated, find the indices of the  $L$  largest  $\beta_k$ s. Let these indices be  $\mathcal{O}_2$ .

**b.** Merge  $\mathcal{O}_1$  and  $\mathcal{O}_2$ , i.e. Let  $\mathcal{O} = \mathcal{O}_1 \cup \mathcal{O}_2$ .

**c.** For all indices in  $k \in \mathcal{O}$  make a decision on  $\hat{x}_k$  (as in Step 5 of Algorithm I). For all indices  $k \notin \mathcal{O}$ , let  $\hat{x}_k = 0$ .

**d.** Identify the indices corresponding to the  $L$  largest (in absolute value) coefficients of  $\hat{\mathbf{x}}$ . Update  $\mathcal{O}_1$  to be this set of  $L$  indices.

**e.** The variable vertices  $k \in \mathcal{O}_1$ , send out their messages as was decided in Step 3 of Algorithm I. The variable vertices  $k \notin \mathcal{O}_1$ , send out their messages with 0 mean and the variance that was decided in Step 3 of Algorithm 1.

**5. Decisions:** Make decisions only the vertices in  $\mathcal{O}$ . Once these are calculated, keep the  $L$  indices with the largest  $|\hat{x}_k|$ ,  $k \in \mathcal{O}$ . Set all other indices to 0.

**6. Iterations:** Repeat (2), (3), (4) and (5) until stopping criterion  $\mathcal{T}$  is reached.

**Output:** The estimate is  $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_M)^T$ .

---

be 0, but the variance (i.e. the uncertainty about the estimate itself) is kept. By modifying the messages this way, we not only enforce sparsity at the final stage, but also throughout the algorithm.

We note that the noise level  $\sigma^2$  is an optional input to the algorithm. Our simulations indicate that the algorithm works without this knowledge also. However, if this extra statistical information is available, it is easily incorporated into the algorithm in a natural way and results in a performance increase.

SuPrEM II has complexity  $O(M)$ . The only significant operation different than those in SuPrEM I is the determination of the largest  $L$  elements of  $\beta$  and  $\hat{\mathbf{x}}$ . This could be done with  $O(M)$  complexity, as described in [15] (Chapter 9). A more straightforward implementation for this stage might use sorting of the relevant coefficients, which would result in a higher complexity of  $O(M \log M)$  for the overall algorithm.

### E. Reweighted Algorithms

For high  $L/N$  ratios, simulation results show that SuPrEM I and SuPrEM II still perform well. However more iterations are needed to achieve very low distortion levels, which may be undesirable. Thus we propose a modification to SuPrEM I and SuPrEM II to speed up the convergence that uses estimates generated within a few iterations. In compressive sensing, employing prior estimates to improve the final solution has been used for  $\ell_1$  approximation [14], but this increases the running time by a factor of reweighting steps.

Next, we motivate for our reweighing approach. In our algorithms, the initial choice of  $\{\beta_k^{(0)} = |(\mathbf{F}^T \mathbf{r})_k|^2 / d_v^2\}$  is based on the intuition that  $\beta_k$  must be proportional to  $|x_k|^2$ . By providing a better estimate for the initial  $\{\beta_k^{(0)}\}$ , the rate of convergence may be improved. The algorithm is initiated with  $\beta^{(0)}$  as above and is run for  $T_{r_1}$  iterations. At the end of this stage, we re-initialize  $\beta^{(0)'}$  to be

$$\beta_k^{(0)'} = |\hat{x}_k^{(T_{r_1})}|^2 + |(\mathbf{F}^T (\mathbf{r} - \mathbf{F} \hat{\mathbf{x}}^{(T_{r_1})}))_k|^2 / d_v^2,$$

and the algorithm is run for  $T_{r_2}$  iterations. This process is repeated recursively until convergence or  $\mathcal{R}$  times. We note that  $\sum_{k=1}^{\mathcal{R}} T_{r_k} = T$ , where  $T$  is the original number of fixed iterations. Thus the total number of iterations remains unchanged when we use reweighing.

#### IV. SIMULATION DETAILS

##### A. Simulation Setup

In our simulations we used LDFs with parameters (3, 6), (3, 12) and (3, 24) for  $M/N = 2, 4, 8$  and  $M = 10000$ . We constructed these frames using the progressive edge growth algorithm [22], avoiding cycles of length 4 when possible<sup>2</sup>. Simulations will be presented for SNR = 12, 24, 36 dB, as well as the noiseless case. For various choices of  $L$  and SNR, we ran 1000 Monte-Carlo simulations for each value, where  $\mathbf{x}$  is generated as a signal with  $L$  non-zero elements that are picked from a Gaussian distribution. The support of  $\mathbf{x}$  is picked uniformly at random. Once  $\mathbf{x}$  is generated, it is normalized such that  $\|\mathbf{F}\mathbf{x}\|_2 = \sqrt{N}$ . Thus  $\text{SNR} = 10 \log_{10} \frac{1}{\sigma^2}$ .

Let  $\mathcal{G}$  be the genie decoder that has full information about  $\text{supp}(\mathbf{x}) = \{i : x_i \neq 0\}$ . Let the output of this decoder be  $\hat{\mathbf{x}}_{\text{genie}} = \mathcal{G}(\mathbf{r})$  obtained by solving the least squares problem involving  $\mathbf{r}$  and the matrix formed by the columns of  $\mathbf{F}$  specified by  $\text{supp}(\mathbf{x})$ . We define the following genie distortion measure:

$$\bar{d}_g(\mathbf{x}, \hat{\mathbf{x}}_{\text{genie}}) = \frac{\|\mathbf{x} - \hat{\mathbf{x}}_{\text{genie}}\|_2^2}{\|\mathbf{x}\|_2^2}.$$

This distortion measure is invariant to the scaling of  $\mathbf{x}$  for a fixed SNR. For any other recovery algorithm that outputs an estimate  $\hat{\mathbf{x}}$ , we let

$$\bar{d}_e(\mathbf{x}, \hat{\mathbf{x}}_e) = \frac{\|\mathbf{x} - \hat{\mathbf{x}}_e\|_2^2}{\|\mathbf{x}\|_2^2},$$

where the subscript  $e$  denotes the estimation procedure. We will be interested in the performance of an estimation procedure with respect to the genie decoder. To this end, we define

$$\mathcal{D}_{e/g}(\mathbf{x}, \hat{\mathbf{x}}_e, \hat{\mathbf{x}}_{\text{genie}}) = \frac{\|\mathbf{x} - \hat{\mathbf{x}}_e\|_2^2}{\|\mathbf{x} - \hat{\mathbf{x}}_{\text{genie}}\|_2^2} = \frac{\bar{d}_e(\mathbf{x}, \hat{\mathbf{x}}_e)}{\bar{d}_g(\mathbf{x}, \hat{\mathbf{x}}_{\text{genie}})}.$$

We will be interested in this quantity averaged over  $K$  Monte-Carlo simulations, and converted to dB. The closer this quantity is to 0 dB means the closer the performance of the estimation procedure is to the performance of the genie decoder.

<sup>2</sup>We also tested LDFs with 4 cycles and this does not seem to have an adverse effect on the average distortion in the presence of noise.

In other cases, such as the noiseless case, we will be interested in the empirical probability of recovery. For  $K$  Monte-Carlo simulations, this is given by

$$P_{\text{rec}} = \frac{1}{K} \sum_{k=1}^K \mathbb{I}(\mathbf{x} \sim \hat{\mathbf{x}}_e),$$

where  $\mathbb{I}(\cdot)$  is the indicator function for  $(\cdot)$  (1 if  $(\cdot)$  is true, 0 otherwise). We will define the relation  $\mathbf{x} \sim \hat{\mathbf{x}}_e$  to be true only if  $\text{supp}(\mathbf{x}) = \text{supp}(\hat{\mathbf{x}}_e)$ , unless otherwise specified.

A number of different stopping criterion can be used for  $T$ : 1)  $\hat{\mathbf{x}}$  converges, 2) The minimum value of  $\{\|\mathbf{r} - \mathbf{F}\hat{\mathbf{x}}^{(t)}\|_2\}_t$  does not change for  $T^d$  iterations, 3) A fixed number of iterations  $T$  is reached. In our simulations we use criterion two with  $T^d = 30$  and  $T = 500$ . These values were chosen to make sure that the algorithms did not stop too prematurely. The message passing schedule  $\mathcal{S}$  is described in detail in Appendix I. Finally, for the reweighted algorithm we use 10 reweighings with  $T_{r_1} = \dots = T_{r_{10}} = T/10$ .

##### B. Simulation Results

Simulation results are presented in Figure 5 for exactly sparse signals.

For comparison to our algorithms, we include results for CoSaMP [30] and  $\ell_1$  based methods [10], [11], [13], [16], [19]. For these algorithms we used partial Fourier matrices as measurement matrices. The choice of these matrices is based on their small storage requirements (in comparison to Gaussian matrices), while still satisfying restricted isometry principles. For CoSaMP, we used 100 iterations of the algorithm (and 150 iterations of Richardson's iteration for calculating least squares solutions). For  $\ell_1$  based methods, we used the L1MAGIC package in the noiseless case. In the noisy case, we used both L1MAGIC, and the GPSR package (with Barzilai-Borwein Gradient Projection with continuation and debiasing). Since these two methods approximately perform the same, we include the results for GPSR here. In the implementation of GPSR we fine-tune the value of  $\tau$  and observe that  $\tau = 0.001 \|\mathbf{F}^T \mathbf{r}\|_\infty$  gives the best performance.

Since the outputs of  $\ell_1$  based methods and SuPrEM I are not sparse, we threshold  $\mathbf{x}$  to its  $L$  largest coefficients and postulate these are the locations of the sparse coefficients. For all methods, we solve the least squares problem involving  $\mathbf{r}$  and the matrix formed by the columns of  $\mathbf{F}$  specified by the final estimate for the locations of the sparse coefficients. For partial Fourier matrices we use Richardson's iteration to calculate this vector, whereas for LDFs we use the LSQR algorithm which also has  $O(M)$  complexity [31].

##### C. Discussion of The Results

The simulation results indicate that the SuPrEM algorithms outperform the other state-of-the-art algorithms. In the low SNR regime (SNR = 12 dB), SuPrEM algorithms and the  $\ell_1$  methods have similar performance. In moderate and high SNR regimes, we see that SuPrEM algorithms significantly outperform the other algorithms both in terms of distortion and in terms of the maximum sparsity they can work at.



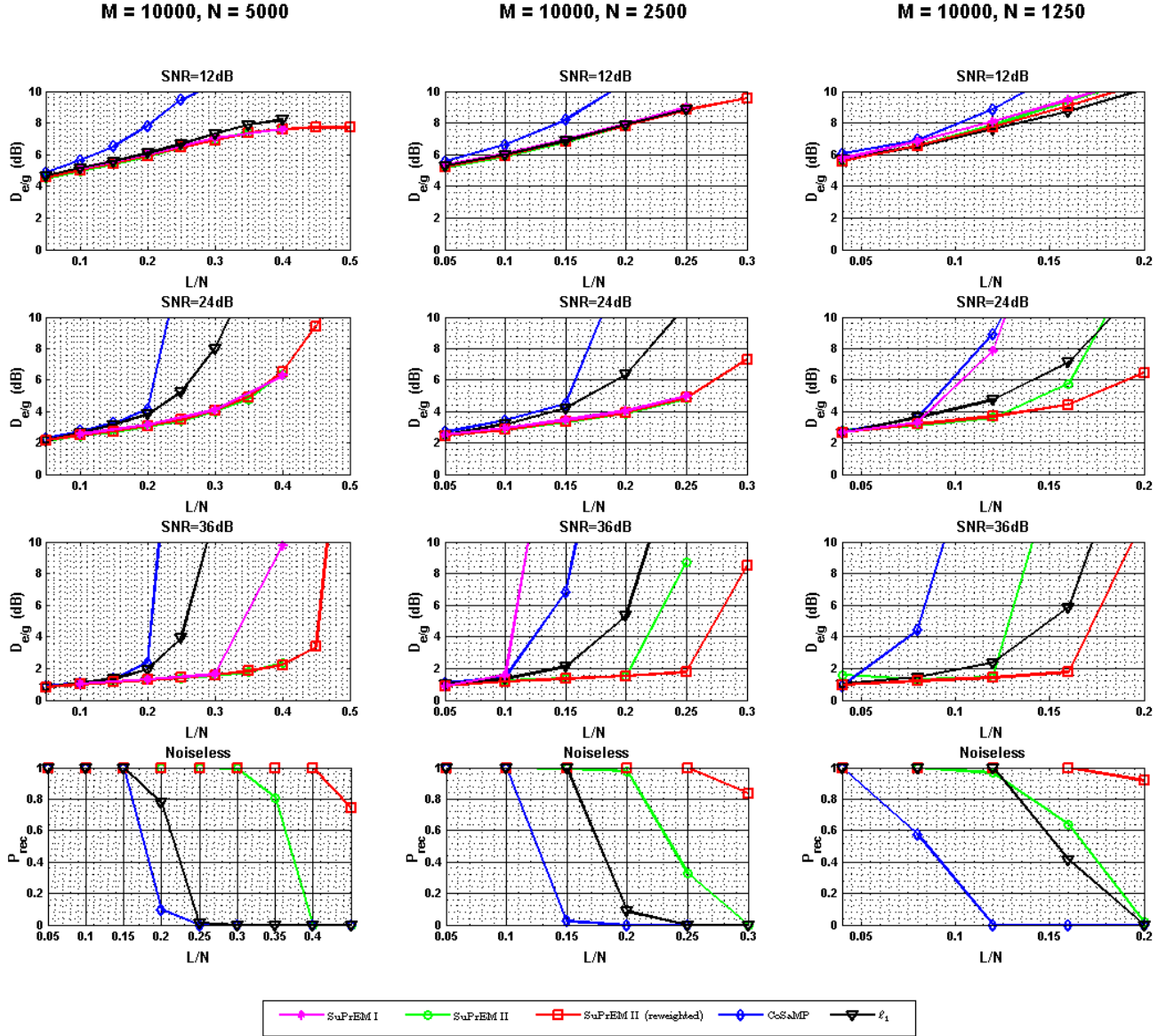


Fig. 5. Performance comparison of recovery algorithms for sparse signals with Gaussian non-zero components.

Furthermore for different values of  $N$ , the maximum sparsity scales as  $L = O(N/\log(M/N))$ , which is the same scaling as those of other methods. As we discussed previously the performance of SuPrEM I degrades as sparsity and SNR increases. We also observe that the reweighted SuPrEM II algorithm outperforms the regular SuPrEM II algorithm, even though the maximum number of iterations are the same.

Finally, compared to the other methods for the noiseless problem, the SuPrEM algorithms can recover signals that have a higher number of non-zero elements. In this case, the reweighted algorithm performs the best, and converges faster. We also note that the results presented for CoSaMP and  $\ell_1$  based methods for the noiseless case are optimistic, since we declare success in recovery if  $\bar{d}_e(\mathbf{x}, \hat{\mathbf{x}}_e) < 10^{-6}$ . We needed to introduce this measure, since these algorithms tend to miss a small portion of the support of  $\mathbf{x}$  containing elements of small magnitude.

We also note that for both partial Fourier matrices and LDFs, the quantity  $\bar{d}_g(\mathbf{x}, \hat{\mathbf{x}}_{genie})$  is almost the same for a fixed  $L$  and SNR. This means that  $\mathcal{D}_{e/g}(\mathbf{x}, \hat{\mathbf{x}}_e, \hat{\mathbf{x}}_{genie})$  provides an objective performance criterion in terms of relative mean-square error with respect to the genie bound, as well as in terms of absolute distortion error  $\bar{d}_e(\mathbf{x}, \hat{\mathbf{x}}_e)$ .

#### D. Simulation Results for Natural Images

For the testing of compressible signals, instead of using artificially generated signals, we used real-world compressible signals. In particular, we compressively sensed the db2 wavelet coefficients of the  $256 \times 256$  (raw) peppers image using  $N = 17000$  measurements. Then we used various recovery algorithms to recover the wavelet coefficients, and we did the inverse wavelet transform to recover the original image.

For SuPrEM algorithms, we used a rate (3, 12) LDF with  $M = 68000$  (the wavelet coefficients vector was padded





Fig. 6. Performance comparison of recovery algorithms with a  $256 \times 256$  natural image whose db2 wavelet coefficients are compressively sensed with  $N = 17000$  measurements.

with zeros to match the dimension). We set  $L = 8000$  (the maximum sparsity the algorithm converged at) for SuPrEM II. We ran the algorithm first with  $\sigma = 0$ . We also accommodated for noise, and estimated the per measurement noise to be  $\sigma = 0.1 \frac{\|\mathbf{r}\|_2}{\sqrt{N}}$  and ran the algorithm again<sup>3</sup>. We ran our algorithms for just 50 iterations. For the reweighted SuPrEM II algorithm, we let  $\sigma = 0$  and we reweighted after 5 steps of the algorithm for a total of 10 reweightings. For SMP, we used the SMP package [7]. We used a matrix generated by this package, and  $L = 8000$ . For the remaining methods, we used partial Fourier matrices whose rows were chosen randomly. For  $\ell_1$

<sup>3</sup>With this value of  $\sigma$ , SuPrEM I also provides a similar performance. However since the output in this case is very similar to that of SuPrEM II, we do not include it in the figure.

with equality constraints, we used the L1MAGIC package. For LASSO, we used the GPSR package and  $\tau = 0.001 \|\mathbf{F}^T \mathbf{r}\|_\infty$ , as described previously, and we thresholded the output to  $L = 8000$  sparse coefficients and solved the appropriate least squares problem to get the final estimate. For CoSaMP and Subspace Pursuit, we used 100 iterations of the algorithm (and 150 iterations for the Richardson's iteration for calculating the least square solutions). For these algorithms, we used  $L = 3000$  for CoSaMP, and  $L = 3500$  for Subspace Pursuit. These are slightly lower than the maximum sparsities they converged at ( $L = 3500$  and  $L = 4000$  respectively), but the values we used resulted in better visual quality and PSNR values. The results are depicted in Figure 6.

The PSNR values for the methods are as follows: 23.41

dB for SuPrEM II, 23.83 dB for SuPrEM II (with non-zero  $\sigma^2$ ), 24.79 for SuPrEM II (reweighted), 20.18 dB for CoSaMP, 19.51 dB for SMP, 21.62 dB for  $\ell_1$ , 23.61 dB for LASSO, 21.27 dB for Subspace Pursuit. Among the algorithms that assume no knowledge of noise, we see that SuPrEM II outperforms the other algorithms both in terms of PSNR value and in terms of visual quality. The two algorithms that accommodate noise, SuPrEM II (in this case SuPrEM I also produces a similar output) and LASSO have similar PSNR values. Finally, the reweighted SuPrEM II also assumes no knowledge of noise, and outperforms all other methods by about 1 dB and also in terms of visual quality, without requiring more running time.

### E. Further Results

We studied the effect of the change of degree distributions. For a given  $M/N$  ratio, we need to keep the ratio of  $d_c/d_v$  fixed however the values can be varied. Thus we compared the performance of  $d_v = 3$  LDFs to  $d_v = 5$  LDFs, and observed that the latter actually performed slightly better. However, having a higher  $d_v$  means more operations are required. We also observed that the number of iterations required for convergence was slightly higher. Thus we chose to use  $d_v = 3$  LDFs that allowed faster decoding. We also note that increasing  $d_v$  too much (while keeping  $M/N$  fixed) results in performance deterioration, since the graph becomes less sparse, and we run into shorter cycles which affect the performance of SPA.

We also tested the performance of our constructions and algorithms at  $M = 100000$ . With  $L/M$  and  $N/M$  fixed, interestingly the performance improves as  $M \rightarrow \infty$  for Gaussian sparse signals for a fixed maximum number of 500 iterations. This is in line with intuitions drawn from Shannon Theory [3]. Another interesting observation is that the number of iterations remain unchanged in this setting. In general, we observed that the number of iterations required for convergence is only a function of  $L/M$  and does not change with  $M$ .

## V. CONCLUSION

In this paper, we constructed an ensemble of measurement matrices with small storage requirements. We denoted the members of this ensemble as Low Density Frames (LDF). For these frames, we provided sparse reconstruction algorithms that have  $O(M)$  complexity and that are Bayesian in nature. We evaluated the performance of this ensemble of matrices and their decoding algorithms, and compared their performance to other state-of-the-art recovery algorithms and their associated measurement matrices. We observed that in various cases of interest, SuPrEM algorithms with LDFs outperformed the other algorithms with partial Fourier matrices. In particular, for Gaussian sparse signals and Gaussian noise, we are within 2 dB range of the theoretical lower bound in most cases.

There are various interesting research problems in this area. One is to find a deterministic message-passing schedule that performs as well as (or better than) our probabilistic message-passing schedule and that is amenable to analysis. Another

open problem is to analyze the performance of the iterative decoding algorithms for the LDFs theoretically, which may in turn lead to useful design tools (like Density Evolution [33]) that might help with the construction of LDFs with irregular degree distributions. Adaptive measurements using the soft information available about the estimates, as well as online decoding (similar to Raptor Codes [37]) is another open research area. Finally, if further information is available about the statistical properties of a class of signals (such as block-sparse signals or images represented on wavelet trees as in [5]), the decoding algorithms may be changed accordingly to improve performance.

## APPENDIX I

### DETAILS ON THE MESSAGE-PASSING SCHEDULE

A message-passing schedule determines the order of messages passed between variable and check nodes of a factor graph. Traditionally, with LDPC codes, the so-called “flooding” schedule is used. In this schedule, at each iteration, all the variable nodes pass messages to their neighboring check nodes. Subsequently, all the check nodes pass messages to their neighboring variable nodes. For a cycle-free graph, SPA with a flooding schedule correctly computes a-posteriori probabilities [9], [49]. An alternative schedule is the “serial” schedule, where we go through each variable node serially and compute the messages to the neighboring nodes. The order in which we go through variable nodes could be lexicographic, random or based on reliabilities.

In this section, we propose the following schedule based on the intuition derived from our simulations and results from LDPC codes [27], [49]: For the first iteration, all the check nodes send messages to variable nodes and vice-versa in a flooding schedule. After this iteration, with probability  $\frac{1}{2}$  each check node is “on” or “off”. If a check node is off, it marks the edges connected to itself as an “inactive”, and sends back the messages it received to the variable nodes. If a check node is on, it marks the edges connected to itself as “active” and computes a new message. At the variable nodes, when calculating the new beta, we only use the information coming from active edges. That is for  $k = 1, 2, \dots, M$ , let  $\{k_1, k_2, \dots, k_{d_v}\}$  be the indices of the check nodes connected to the  $k^{\text{th}}$  variable node  $x_k$ . Let the incoming message from the check node  $r_{k_j}$  to the variable node  $x_k$  at the  $t^{\text{th}}$  iteration be  $(\mu_{k_j}^{(t)}, \nu_{k_j}^{(t)})$  for  $j = 1, \dots, d_v$ . We will have

$$\lambda_k^{(t)} = \left( \sum_{(k, k_j) \text{ is an active edge}} \frac{1}{\nu_{k_j}^{(t)}} + \frac{1}{\beta_k^{(t-1)}} \right)^{-1},$$

$$\mu_k^{(t)} = \lambda_k^{(t)} \left( \sum_{(k, k_j) \text{ is an active edge}} \frac{\mu_{k_j}^{(t)}}{\nu_{k_j}^{(t)}} \right),$$

and

$$\beta_k^{(t)} = \frac{(\mu_k^{(t)})^2 + \lambda_k^{(t)}}{3}.$$

Thus when there is no active edge, we do not perform a  $\beta$  update. For the special case when there is only one active edge  $(k, k_j)$ , we let  $\mu_k^{(t)} = \mu_{k_j}^{(t)}$ . This is because the intrinsic

information is more valuable, and the estimate on  $\beta_k^{(t-1)}$  tends to be not as reliable. When we calculate the point estimate, we use all the information at the node, including the reliable and unreliable edges, i.e.

$$\hat{V}_k^{(t)} = \left( \sum_{j=1}^{d_v} \frac{1}{\nu_{k_j}^{(t)}} + \frac{1}{\beta_k^{(t)}} \right)^{-1},$$

$$\hat{x}_k^{(t)} = \hat{V}_k^{(t)} \left( \sum_{j=1}^{d_v} \frac{\mu_{k_j}^{(t)}}{\nu_{k_j}^{(t)}} \right).$$

It is noteworthy that the flooding schedule and serial schedules tend to converge to local minima and they do not perform as well as this schedule we proposed.

## REFERENCES

- [1] M. Akçakaya and V. Tarokh, "A Frame Construction and A Universal Distortion Bound for Sparse Representations," *IEEE Trans. Sig. Proc.*, vol. 56, pp. 2443-2550, June 2008.
- [2] M. Akçakaya and V. Tarokh, "On Sparsity, Redundancy and Quality of Frame Representations," IEEE Int. Symposium on Information Theory (ISIT), Nice, France, June 2007.
- [3] M. Akçakaya and V. Tarokh, "Shannon theoretic limits on noisy compressive sampling," arXiv:0711.0366v1 [cs.IT], Nov. 2007.
- [4] D. Andrews and C. Mallows, "Scale mixtures of normal distributions," *J. R. Stat. Soc.*, vol. 36, pp. 99 - 102, 1974.
- [5] R. Baraniuk, V. Cevher, M. Duarte, and C. Hegde, "Model-based compressive sensing," arXiv:0808.3572v2, Sept. 2008.
- [6] R. Berinde, A. C. Gilbert, P. Indyk, H. Karloff, and M. J. Strauss, "Combining geometry and combinatorics: A unified approach to sparse signal recovery," preprint, 2008.
- [7] R. Berinde, P. Indyk, and M. Ružić, "Practical near-optimal sparse recovery in the  $\ell_1$  norm," Proc. Allerton Conference on Communication, Control, and Computing, Monticello, IL, September 2008.
- [8] J. M. Bioucas-Dias, "Bayesian Wavelet-Based Image Deconvolution: A GEM Algorithm Exploiting a Class of Heavy-Tailed Priors," *IEEE Trans. Image Proc.*, vol. 15, pp. 937-951, Apr. 2006.
- [9] C. M. Bishop, *Pattern Recognition and Machine Learning*, First Edition, Springer, New York, NY, 2006.
- [10] E. J. Candès, J. Romberg, "Practical signal recovery from random projections," presented at the *Wavelet Appl. Signal Image Process. XI, SPIE Conf.*, San Diego, CA, 2005.
- [11] E. J. Candès, J. Romberg, T. Tao, "Stable signal recovery for incomplete and inaccurate measurements," *Commun. Pure Appl. Math.*, vol. 59, pp. 1207-1223, Aug. 2006.
- [12] E. J. Candès and T. Tao, "The Dantzig selector: statistical estimation when  $p$  is much larger than  $n$ ," *Annals of Statistics*, 35, pp. 2313-2351, Dec. 2007.
- [13] E. J. Candès, T. Tao, "Decoding by Linear Programming," *IEEE Trans. Inf. Theory*, vol. 51, pp. 4203-4215, Dec. 2005.
- [14] E. J. Candès, M. Wakin and S. Boyd, "Enhancing sparsity by reweighted  $\ell_1$  minimization," *J. Fourier Anal. Appl.*, vol. 14, pp. 877-905.
- [15] T. Cormen, C. Leiserson, L. Rivest, and C. Stein, *Introduction to Algorithms*, Second Edition, MIT Press, Cambridge, MA, 2001.
- [16] D. L. Donoho, "Compressed Sensing," *IEEE Trans. Inf. Theory*, vol. 52, pp. 1289-1306, April 2006.
- [17] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing: Closing the gap between performance and complexity," arXiv:0803.0811v2 [cs.NA], March 2008.
- [18] M. A. T. Figueiredo and R. Nowak, "Wavelet-based image estimation: An empirical bayes approach using Jeffreys noninformative prior," *IEEE Trans. Image Proc.*, vol. 10, pp. 1322-1331, Sep. 2001.
- [19] M. A. T. Figueiredo, R. D. Nowak and S. J. Wright, "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, pp. 586-598, Dec. 2007.
- [20] A. K. Fletcher, S. Rangan and V. K. Goyal, "Necessary and Sufficient Conditions on Sparsity Pattern Recovery," arXiv:0804.1839v1 [cs.IT], Apr. 2008.
- [21] R. G. Gallager, *Low-Density Parity-Check Codes*, MIT Press, Cambridge, MA, 1963.
- [22] X.-Y. Hu, E. Eleftheriou and D. M. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, pp. 386-398, Jan. 2005.
- [23] S. Ji, Y. Xue and L. Carin, "Bayesian compressive sensing," *IEEE Trans. on Sig. Proc.*, vol. 56, pp. 2346-2356, June 2008.
- [24] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor Graphs and the Sum-Product Algorithm," *IEEE Trans. Inf. Theory*, vol. 47, pp. 498-519, Feb. 2001.
- [25] D. J. C. MacKay, "Good error correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, pp. 399-431, Mar. 1999.
- [26] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*, First Edition, Cambridge University Press, Cambridge, UK, 2002.
- [27] Y. Mao and A. H. Banihashemi, "Decoding Low-Density Parity-Check Codes With Probabilistic Scheduling," *IEEE Comm. Letters*, vol. 5, pp. 414-416, Oct. 2001.
- [28] G. J. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*, First Edition, John Wiley & Sons, New York, NY, 1997.
- [29] T. K. Moon, "The EM algorithm in signal processing," *IEEE Sig. Proc. Mag.*, vol. 13, pp. 47-60, Nov. 1996.
- [30] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," arXiv:0803.2392v2 [math.NA], Apr. 2008.
- [31] C. C. Paige and M. A. Saunders, "LSQR: Sparse Linear Equations and Least Squares Problems," *ACM Transactions on Mathematical Software (TOMS)*, vol. 8, pp.195-209, June 1982.
- [32] J. Portilla, V. Strela, M. J. Wainwright and E. P. Simoncelli, "Image Denoising Using Scale Mixtures of Gaussians in the Wavelet Domain," *IEEE Trans. Image Proc.*, vol. 12, pp. 1338-1351, Nov. 2003.
- [33] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599-618, Feb. 2001.
- [34] C. Robert, *The Bayesian Choice: A Decision Theoretic Motivation*, First Edition, New York, NY, Springer-Verlag, 1994.
- [35] S. Sarvotham, D. Baron, and R. Baraniuk, "Sudocodes - Fast Measurement and Reconstruction of Sparse Signals," *Proc. IEEE Int. Symp. on Inf. Theory (ISIT)*, Seattle, WA, July 2006.
- [36] S. Sarvotham, D. Baron, and R. Baraniuk, "Compressed Sensing Reconstruction via Belief Propagation," preprint, 2006.
- [37] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, pp. 2551-2567, June 2006.
- [38] M. Sipser and D. A. Spielman, "Expander codes," *IEEE Trans. Inf. Theory*, vol. 42, pp. 1710-1722, Nov. 1996.
- [39] R. M. Tanner, "A Recursive Approach to Low Complexity Codes," *IEEE Trans. Inf. Theory*, vol. 27, pp. 533-547, Sept. 1981.
- [40] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, vol. 1, pp. 211-244, 2001.
- [41] M. E. Tipping, "Bayesian inference: An introduction to principles and practice in machine learning," in O. Bousquet, U. von Luxburg, and G. Rtsch (Eds.), *Advanced Lectures on Machine Learning*, pp. 41-62, Springer, 2004.
- [42] J. A. Tropp, "Topics in Sparse Approximation", Ph.D. dissertation, Computational and Applied Mathematics, UT-Austin, August 2004.
- [43] J. A. Tropp, "Just relax: Convex programming methods for identifying sparse signals", *IEEE Trans. Inf. Theory*, vol. 51, no. 3, pp. 1030-1051, Mar. 2006.
- [44] J. A. Tropp, A. C. Gilbert, "Signal recovery from partial information via Orthogonal Matching Pursuit," *IEEE Trans. Inf. Theory*, vol. 53, pp.4655-4666, Dec. 2007.
- [45] M. J. Wainwright, "Information-Theoretic Limits on Sparsity Recovery in the High-Dimensional and Noisy Setting," Technical Report, UC Berkeley, Department of Statistics, Jan. 2007.
- [46] M. J. Wainwright, "Sharp thresholds for noisy and high-dimensional recovery of sparsity using  $\ell_1$ -constrained quadratic programming," Technical report, UC Berkeley, Department of Statistics, May 2006.
- [47] Y. Weiss and W. T. Freeman, "Correctness of belief propagation in Gaussian graphical models of arbitrary topology," *Proc. Adv. Neural Inform. Processing Syst.*, vol. 12, Dec. 1999.
- [48] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Linköping University, Sweden, 1996.
- [49] H. Xiao and A. H. Banihashemi, "Graph-Based Message-Passing Schedules for Decoding LDPC Codes," *IEEE Trans. on Comm.*, vol. 52, pp. 2098-2105, Dec. 2004.
- [50] W. Xu and B. Hassibi, "Efficient compressive sensing with deterministic guarantees using expander graphs," *Proc. IEEE Inf. Theory Workshop*, Lake Tahoe, CA, Sept. 2007.